Compas'2016 : Parallélisme/ Architecture / Système
Lorient, France, du 5 au 8 juillet 2016

Filament : a cohort construction service for decentralized collaborative editing platforms

A. C. Resmi & François Taiani

Université de Rennes 1 - IRISA, Rennes, France
{rariyatt, francois.taiani}@irisa.fr

Résumé
Distributed collaborative editors allow several remote users to contribute concurrently to the same document. Unfortunately, the currently deployed editors can only support a limited number of concurrent users. A number of peer-to-peer solutions have therefore been proposed to remove this limitation and allow a large number of users to work collaboratively. These approaches however tend to assume that all users edit the same set of documents, which is unlikely to be the case if such systems should become widely used and ubiquitous. In this paper we discuss a novel cohort-construction approach that allow users editing the same documents to rapidly find each other.

Mots-clés : Collaborative editing, Overlay networks, Decentralized systems, DHT

1. Introduction

A new generation of low-cost computers known as *plug computers* has recently appeared, offering users the possibility to create decentralized federation of domestic servers to host collaborative services (such as collaborative editing, data sharing). These federations differ from traditional cloud computing in the autonomy they grant to their participants. They also differ from P2P networks as participants tend to remain stable, producing little churn.
Several promising contributions [4, 16, 23] for decentralized peer-to-peer collaborative editing have already been proposed that can be ported to federations of plug computers. Most of the works generally assume that all nodes in the system edit the same document, or the same set of documents. In practice if we consider a very large set of users (e.g. several millions), users editing the same document need to find each other first in order to build a clique of interested users. A DHT [19, 21, 24] can be used to this aim, by using documents as key, and appending new nodes to set nodes already editing the document. However the choice of DHT is not optimal, as it adds an additional indirection in finding the actual editor of a document, and create potential hot-spot for nodes handling highly popular documents. Furthermore there will typically be fewer documents than nodes, in contrast to a typical DHT, which is designed to handle more keys than nodes.
In this paper we propose *Filament*, an approach that eliminates the need to a DHT, and allows nodes editing the same document to find each other by generating a *routing field* around themselves. The remainder of the paper is organized as follows. We first present the problem we address and our intuition (Sec. 2) ; we then present our algorithm (Sec. 3), and its evaluation (Sec. 4). We finally discuss related work (Sec. 5), and conclude (Sec. 6).

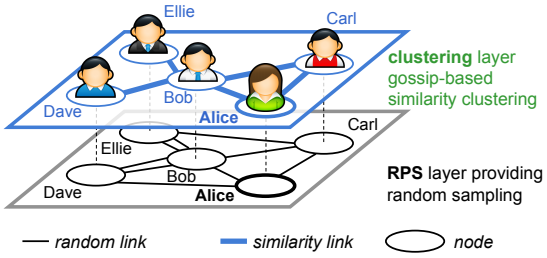

FIGURE 1 – Overlay Architecture

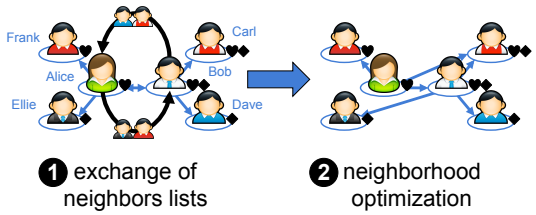

FIGURE 2 – P2P neighborhood optimization

2. Background, Problem, and Intuition

2.1. Collaborative editing and cohort construction

Most of the deployed distributed collaborative editors are centralized, hosted in tightly integrated environments and show poor scalability [1, 2]. To overcome this limitation, several promising works have been proposed to host collaborative editing platforms in decentralized peer-to-peer architectures [4,16,23]. Most of these approaches assume that all users in a system edit the same document. In a large community, this assumption is unrealistic, and users editing the same document need a mechanism to find each other. This is a particular case of peer-to-peer search, which has been extensively researched in the past both in unstructured [6–8,13,17] and structured systems, in particular in DHT [19–21, 24]. Unstructured approaches have probabilistic guarantees : a resource might be present in the system, but it may not get found.
Structured approaches such as DHTs typically have deterministic guaranties, but assume that the number of items to be stored is much higher than the number of storage nodes available. This is in stark contrast to distributed collaborative platforms, in which the number of documents being edited is smaller than the number of users. Furthermore, these systems use consistent hashing techniques in which a node's role in the system is independent of this node's particular interests (in our case here documents), thus adding an additional layer of redirection.
To address these challenges, we propose a novel decentralized service that connects together users interested in the same document without relying on the additional indirection implied by DHTs, while delivering deterministic guarantees, contrary to the unstructured networks.

2.2. Self-organising overlays

Our proposal, called Filament, composes together several self-organising overlay networks to deliver its service. Overlay networks connect computers (aka *nodes*) on top of a standard point-to-point network (e.g. TCP/IP) in order to add additional properties and services to this underlying network. Self-organising overlay networks are a special type of such overlays, in which nodes organize themselves according to some similarity function so that, once the system has converged, each nodes n is connected to its k closest neighbours (Fig. 1 and 2).

3. System

In a large CE system, users editing the same document need to find each other in order to propagate modifications between themselves. The approach we propose allows nodes to locate each other by generating a *routing field* based on a set of self-organising overlays.

3.1. System model

We consider a network consisting of a large number of nodes $\mathcal{N} = \{n_1, n_2, .., n_N\}$. The network is dynamic : nodes may join or leave at anytime. Nodes are assigned unique identifiers and com-

TABLE 1 – Notations and Entities

$n.id$	node identifier of node n
k_n	number of documents being edited by node n
$n.\mathcal{D}$	list of documents edited by node n depicted as $\{d_1^n, d_2^n, ..., d_{k_n}^n\}$
$n.\mathcal{H}$	helper overlay associated with node n
$n.\mathcal{F}$	fingers of node n
$n.view(d_p)$	set of collaborators for document d_p contained in node n
$F_n[i]$	node in the finger list which is at a distance of i from node n
l	maximum size of the collaborators list associated with each document in a node
lh	size of the helper overlay
lf	size of finger list

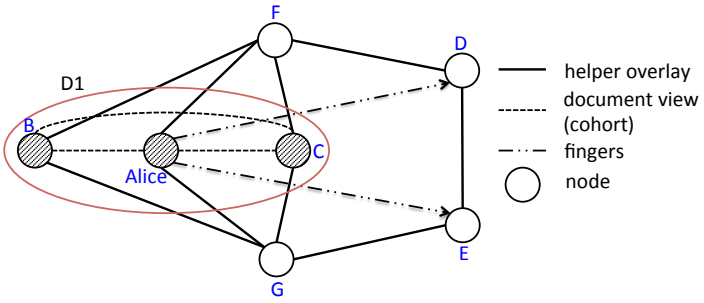

FIGURE 3 – Overlay view

municate using messages. We assume that nodes are connected through an existing network, such as the Internet, allowing every node to potentially communicate with any other node as soon as it knows this other node's identifier. Nodes are organized in an overlay network in which each node knows the identifiers of a set of other nodes, which form its *neighborhood*.

This neighborhood can change over time to fulfil the system's objectives. Assuming a collaborative editing environment we can imagine each node to be a user. Each node/user n may be editing a set of zero or more documents (noted $n.\mathcal{D}$) at any given time according to their interests.

3.2. Filament

All the nodes in the system are part of several sub-overlays as shown in Fig. 3. There is a helper overlay ($\mathcal{H}$) associated with each node. This helper overlay is filled initially using random peers taken from the Random Peer Sampling layer (RPS). As the simulation progresses, $n.\mathcal{H}$ is progressively filled with nodes that have similar but not identical interests. Each node further maintains a specific view for each document it is currently editing, in order to rapidly propagate edits : the aim of Filament is to fill this view as rapidly as possible. In the figure, nodes Alice, B and C will form a document overlay as all of them are editing document D1. In addition to this, each node maintains a set of *fingers*($\mathcal{F}$), which acts as long distance links. Similar to a traditional ring-based DHTs, these links help rapidly locate collaborating nodes, and avoid disjoint partitions. A simplistic view of the system model is shown in Fig. 4. Table 1 summarizes the notations that we use in this paper.

The basic algorithm behind our approach is shown in Fig. 5. The algorithm shows what happens in each round at node n. What we aim to do is to fill the document views associated with each node in a fast manner so as to allow the propagation of edits between collaborating nodes. In addition to this we also need mechanisms that help us in routing i.e. in cases where a new

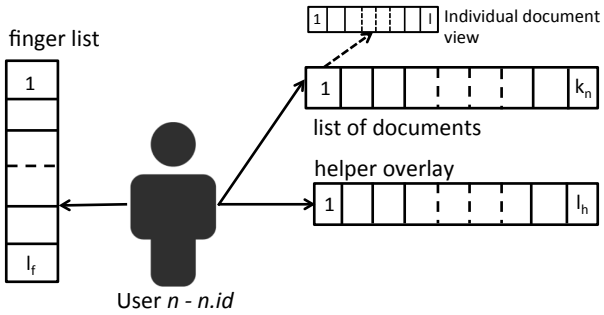

FIGURE 4 – Illustration of the system model

node joins the system or in cases where a new document is added to a node in the system.
In each cycle, a node exchanges its view with the nodes in its helper overlay and also one randomly selected node in the system. A random entry is added with the hope that the system converges faster. Once a node gets the views associated with other nodes, it modifies its own document views and helper overlay. If the other node has the same document, it can make use of the view of that node for getting a better document view for itself. The randomly filled helper overlay is also modified as the simulation progresses so as to fill it with nodes similar to themselves but non-identical. The similarity metric we use here is based on document ids and is novel in that it removes identical nodes. The closer the document ids contained in a pair of nodes, more similar they are to each other. As a node encounters a new node which is not in its views it calculates the similarity and make use of it to edit its finger list and also the helper overlay. The finger list is similar to the fingers in chord. It is an array of nodes where the i-th entry will be a node whose similarity with the given node is i. This helps in providing long distance routing links. The finger lists are mainly used in cases where a node needs to find collaborators for a newly added document. A node can look in its finger list in order to find someone editing the newly added document or to find some one who might be editing a document similar to the newly added document.
As shown in Fig. 5, the procedure $\Delta(n, u)$ returns the similarity value between node n and node u. The procedure $fingers(n, p)$ helps in updating the finger list associated with a node. The system is assumed to be converged when it is able to complete its document views by obtaining atleast l collaborators for each document or all the collaborators for a document which is being edited by less than l users.

4. Evaluation

All the results (Figs. 6-8) are computed with Peersim [14] and are averaged over 10 experiments. The source code is made available in http ://armi.in/resmi/ce1.zip. The results shown in this section are for a base case. The network size is taken as 2^{12} in this scenario. The value of l (document view) and l_h (size of the help overlay) is taken as 10 in all the experiments. Some of the results obtained during the preliminary evaluation are shown here.
Fig. 6 shows the cumulative frequency distribution of converged nodes in each round. It takes about 11 rounds for the system to converge. By convergence we mean that the document views associated with all the nodes are filled. Fig. 7 depicts the case when a new document is added to a node after system convergence which in this case is 11 rounds. As shown in the plot, the node is able to find l collaborators for the newly added document in 4.1 rounds. Fig. 8 depicts how our approach fares compared to a DHT when a node has to find collaborators for a newly

```
1: In round(r) do
2:     r ← random node from 𝒩
3:     forall q ∈ r ∪ n.ℋ
4:         globalview.n ← n.ℋ ∪ ⋃_{d∈n.𝒟} n.view(d)
5:         globalview.q ← q.ℋ ∪ ⋃_{d∈q.𝒟} q.view(d)
6:         n pulls globalview.q
7:         forall d ∈ n.𝒟 ∩ q.𝒟 : n.view(d) ← n.view(d) ∪ q.view(d) truncated to l entries
8:         forall p ∈ globalview.q \ globalview.n
9:             Call Δ(n, p) and update n.ℋ to contain lh closest neighbours
10:            forall d ∈ n.𝒟 ∩ p.𝒟 : n.view(d) ← n.view(d) ∪ p.view(d) truncated to l entries
11:            Call fingers(n, p)
12: Procedure Δ(n, u)
13:     S1 ← n.𝒟 \ u.𝒟 ; S2 ← u.𝒟 \ n.𝒟
14:     S3 ← S1 × S2
15:     m ← min(|x − y|)∀(x, y) ∈ S3
16:     return m
17: Procedure fingers(n, p)
18:     i ← Δ(n, p)
19:     Fn[i] ← p
```

FIGURE 5 – Filament

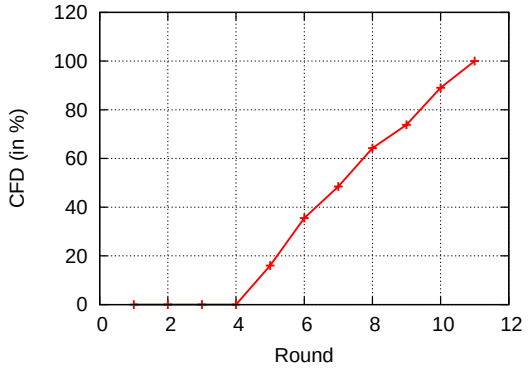

FIGURE 6 – CFD of converged nodes in each round

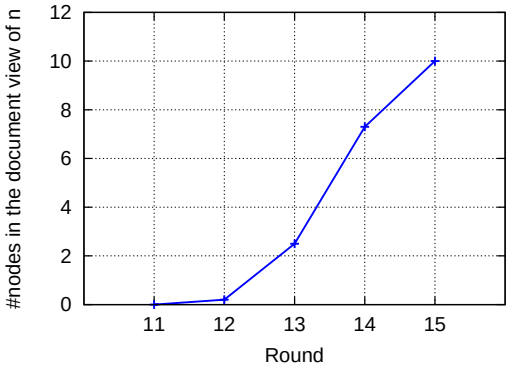

FIGURE 7 – System behaviour when a new document is added to node n

added document. Document latency is the average number of rounds it takes for a node to find collaborators for the new document that is added to it. Here the system is assumed to be converged before the addition of the new document.

5. Related Work

Researchers have been looking into peer-to-peer collaborative editing platforms [4, 9, 11, 12, 16, 23] for some time. Most of these approaches in decentralized peer-to-peer collaborative editing assume that all users in a system participate in the same edition. But in a very large community of users, it may not necessarily be the case. This leads to the problem of finding nodes working on the same document. This kind of search in peer-to-peer system has been extensively researched in the past in both unstructured [6–8, 13, 17] and structured overlays, in particular in the context of *Distributed Hash Tables* DHTs [19–21, 24]. Unstructured approaches have probabilis-

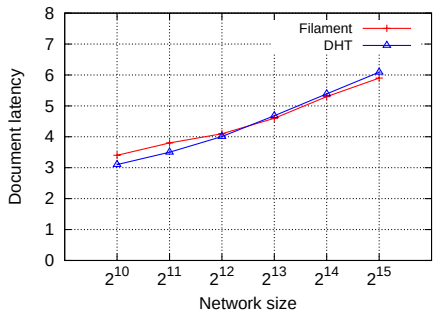

FIGURE 8 – Filament vs DHT based on document latency

tic recall rate which means that they do not provide any deterministic guarantees : a resource might be present in the system, but it may not get found, unless a flooding or exhaustive multicast strategy is used, which might be very costly in massive systems.
Structured approaches such as DHTs typically provide deterministic guaranties, but usually assume that the number of items to be stored is much higher than the number of storage nodes available. Furthermore, these systems use consistent hashing techniques in which a node's role in the system is independent of this node's particular interests. In case of a highly frequented resource, DHTs use load-balancing techniques [10, 18] that typically use virtual nodes or modify the hashing function [5, 7] to spread the load more evenly. These functions are however reactive, and well suited for highly frequented items. The main problem with DHT is that it adds an additional indirection.
Constructing and maintaining the overlay required by a DHT can be a complex error-prone operation. In the past researchers have therefore proposed to employ gossip based [3] self-organisation protocols such T-Man [15] or Vicinity [22] to realize this task.

6. Conclusion and Future Work

Distributed collaborative editors allow several remote users to contribute concurrently to the same document. Deployed editors unfortunately can only support a limited number of concurrent users. A number of peer-to-peer solutions have therefore been proposed to remove this limitation and allow a large number of users to work collaboratively. These approaches however tend to assume that all users edit the same set of documents, which is unlikely to be the case if such systems should become widely used and ubiquitous. In this paper we discuss a novel cohort-construction approach that allow users editing the same documents to rapidly find each other. Our proposal utilises the semantic relations between peers to construct a set of self-organising overlays which we use to route search requests. On the basis of the preliminary evaluation we have conducted so far, we believe the resulting protocol is efficient, scalable, and provides beneficial load-balancing properties over the involved peers.

Acknowledgments

This work was partially funded by the DeSceNt project granted by the Labex CominLabs excellence laboratory of the French Agence Nationale de la Recherche (ANR- 10-LABX-07-01).

Bibliographie

1. Etherpad.

2. Google docs.
3. Bertier (M.), Frey (D.), Guerraoui (R.), Kermarrec (A.-M.) et Leroy (V.). – The gossple anonymous social network. – In *Middleware'10*.
4. Davoust (A.), Skaf-Molli (H.), Molli (P.), Esfandiari (B.) et Aslan (K.). – Distributed wikis : a survey. *Concurrency and Computation : Practice and Experience*, vol. 27, n11, 2015, pp. 2751–2777.
5. DeCandia (G.), Hastorun (D.), Jampani (M.), Kakulapati (G.), Lakshman (A.), Pilchin (A.), Sivasubramanian (S.), Vosshall (P.) et Vogels (W.). – Dynamo : amazon's highly available key-value store. – In *SOSP'07*.
6. Dorrigiv (R.), Lopez-Ortiz (A.) et Prałat (P.). – Search algorithms for unstructured peer-to-peer networks. – In *Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on*, pp. 343–352. IEEE, 2007.
7. Filali (I.) et Huet (F.). – Dynamic ttl-based search in unstructured peer-to-peer networks. – In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pp. 438–447. IEEE, 2010.
8. Gkantsidis (C.), Mihail (M.) et Saberi (A.). – Random walks in peer-to-peer networks : Algorithms and evaluation. *Perform. Eval.*, vol. 63, n3, mars 2006, pp. 241–263.
9. Gupta (A.), Sahin (O. D.), Agrawal (D.) et Abbadi (A. E.). – Meghdoot : content-based publish/subscribe over p2p networks. – In *Middleware'04*.
10. Karger (D. R.) et Ruhl (M.). – Simple efficient load balancing algorithms for peer-to-peer systems. – In *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pp. 36–43. ACM, 2004.
11. Kermarrec (A.-M.) et Triantafillou (P.). – Xl peer-to-peer pub/sub systems. *ACM Computing Surveys (CSUR)*, vol. 46, n2, 2013.
12. Lakshman (A.) et Malik (P.). – Cassandra : a decentralized structured storage system. *SIGOPS Oper. Syst. Rev.*, vol. 44, n2, 2010.
13. Lv (Q.), Cao (P.), Cohen (E.), Li (K.) et Shenker (S.). – Search and replication in unstructured peer-to-peer networks. – In *Proceedings of the 16th international conference on Supercomputing*, pp. 84–95. ACM, 2002.
14. Montresor (A.) et Jelasity (M.). – Peersim : A scalable p2p simulator. – In *P2P 2009*.
15. Montresor (A.), Jelasity (M.) et Babaoglu (O.). – Chord on demand. – In *P2P 2005*.
16. Oster (G.), Mondéjar (R.), Molli (P.) et Dumitriu (S.). – Building a collaborative peer-to-peer wiki system on a structured overlay. *Computer Networks*, vol. 54, n12, 2010, pp. 1939–1952.
17. Otto (F.) et Ouyang (S.). – Improving search in unstructured p2p systems : Intelligent walks (i-walks). – In *Proceedings of the* 7^{th} *International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'06)*, *Lecture Notes in Computer Science*, volume 4224, pp. 1312–1319. Springer, septembre 2006.
18. Rao (A.), Lakshminarayanan (K.), Surana (S.), Karp (R.) et Stoica (I.). – Load balancing in structured p2p systems. *In : Peer-to-Peer Systems II*, pp. 68–79. – Springer, 2003.
19. Ratnasamy (S.), Francis (P.), Handley (M.), Karp (R.) et Shenker (S.). – A scalable content-addressable network. *SIGCOMM Comput. Commun. Rev.*, vol. 31, n4, 2001, pp. 161–172.
20. Rowstron (A. I. T.) et Druschel (P.). – Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. – In *Middleware*, pp. 329–350, 2001.
21. Stoica (I.), Morris (R.), Karger (D.), Kaashoek (M. F.) et Balakrishnan (H.). – Chord : A scalable peer-to-peer lookup service for internet applications. – In *SIGCOMM '01*.
22. Voulgaris (S.) et Steen (M. v.). – Epidemic-style management of semantic overlays for content-based searching. – In *Euro-Par'05*.
23. Weiss (S.), Urso (P.) et Molli (P.). – Logoot-undo : Distributed collaborative editing system

on P2P networks. *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, n8, 2010, pp. 1162–1174.

24. Zhao (B.), Kubiatowicz (J.) et Joseph (A.). – Tapestry : An infrastructure for fault-tolerant wide-area location and routing. *Computer*, vol. 74, 2001.